Interoperability in Software Defined Networking

A Thesis Submitted In Partial Fulfilment of the Requirements for the Degree of

Bachelor of Technology
In
Electronics and Communication Engineering

By
VISHAL MISHRA (111EC0179)

Department of Electronics and Communication Engineering
National Institute of Technology, Rourkela
2015

Interoperability in Software Defined Networking

A Thesis Submitted In Partial Fulfilment of the Requirements for the Degree of

Bachelor of Technology
In
Electronics and Communication Engineering

Under the guidance of
Prof. Santos Kumar Das

By

VISHAL MISHRA (111EC0179)

Department of Electronics and Communication Engineering
National Institute of Technology, Rourkela
2015

DECLARATION

I hereby certify that

1) The work written in this thesis is original and was done by me under the guidance of my supervisor.
2) The work has not been submitted to any other Institute for any degree or diploma.
3) I have followed the guidelines provided by the Institute in preparing the thesis.
4) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
5) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

Vishal Mishra

NATIONAL INSTITUTE OF TECHNOLOGY
ROURKELA

CERTIFICATE

This is to certify that the work in the thesis entitled, "**Interoperability in Software Defined Networking**" submitted by **VISHAL MISHRA** is a record of an original research work carried out by him during the session 2014-2015 under my supervision and guidance, in partial fulfilment of the requirement for the award of Bachelor of Technology Degree in Electronics & Communication Engineering, National Institute of Technology, Rourkela. Neither this thesis nor any part of it has been submitted for any degree or diploma elsewhere.

Prof. Santos Kumar Das
Department of Electronics & Communication
NIT Rourkela
Project Supervisor

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Santos Kumar Das, my guide on this project, for his expert knowledge, guidance and constant support during the whole duration of the project work. Without him, my project would not have been successfully completed. He was always there in time of need and whenever I had any problem related to the project work. Although he had an extremely busy schedule, he always found time to help me. The pivotal thing is that he always motivated me to put in my best efforts and kept pushing me for more.

I would like to extend my deep gratitude to my esteemed professors - Prof. K. K. Mahapatra, Prof. S. M. Hiremath, Prof. U. K. Sahoo, Prof Samit Ari and other staff members for their invaluable feedback and suggestions that helped me in improving my work and also for letting me use the Laboratory facilities whenever I required.

I would like to thank my institution and all the faculty members of ECE department for their help and guidance. They have been a great source of inspiration to me and I thank them from the bottom of my heart.

I am grateful to Phd and M. Tech Research Scholars for their co-operation in usage of laboratories and to all my friends who in some way or other have helped me with this project.

Last but not the least, I would like to thank my parents and well-wishers who have been a constant source of inspiration and motivation for me.

VISHAL MISHRA

ABSTRACT

Today's world emphasizes a lot on speed, performance and ease of use. Taking these demands into consideration it has become mandatory to change all existing systems and transform them into ones that can deliver these characteristics. One such system has been network architecture. Since many years vendors have been creating and developing their own Network Infrastructure, with different policies and different instruction sets. But communication between networks has taken up importance so as to reduce the timing taken by the data packet to pass through the security settings of another network and vice versa. Hence here is where Interoperability comes into play. If we can take all the networks and have a common controller for all the networks, it would reduce the time and complexity for communicating between them. A simple way to achieve it is through Software Defined Networking, where we can use the software to control the flow of data packets as well as monitor the network. But it is not so simple, interoperability has its own sets of issues like cost overhead, security and safety, speed and most importantly acceptability. Network vendors have been apprehensive about accepting SDN because of these concerns that plague it.

This thesis looks at what SDN is and what are the several benefits of it. At the same time looking more into the Interoperability methods and how security concern can be alleviated. This is done by detecting and removing ARP spoofing from the network which is a major concern in modern day networks. Also network monitoring methods are seen which can be used for network speed optimization.

Keywords: *SDN, Interoperability, ARP, OpenFlow*

Contents

Table of Figures

List of Abbreviations

SDN – Software Defined Networking

NOS-Network Operating System

MITM – Man in the Middle

ARP – Address Resolution Protocol

IT – Information Technology

PC – Personal Computer

LAN – Local Area Network

WAN – Wide Area Network

OS – Operating System

VM – Virtual Machine

CPU – Central Processing Unit

DBMS – Database Management System

API – Application Program Interface

OSPF - Open Shortest Path First

MAC - Macintosh

ONF - Open Networking Foundation

SSH – Secure Shell

CHAPTER 1

Introduction

"In a traditional network, developing a new application and modifying the behaviour of any existing devices is a very difficult work. SDN or Software Defined networking overcomes these problems. It does so by shifting the control logic from a device to the centralized place. This shifted control logic is known as SDN Controller or NOS (Network Operating System). The controller can have the global view of an entire network, so we can manage the functionality of any network by using SDN in an efficient manner. OpenFlow, which is a standard protocol, is used to provide the communication between a dumb device and the controller."

The Objective of Interoperability is:

- Ability to run multiple virtual networks.
- Each has a separate control and data plane.
- Coexist together on top of one physical network.
- Can be managed by individual parties that potentially don't trust each other.

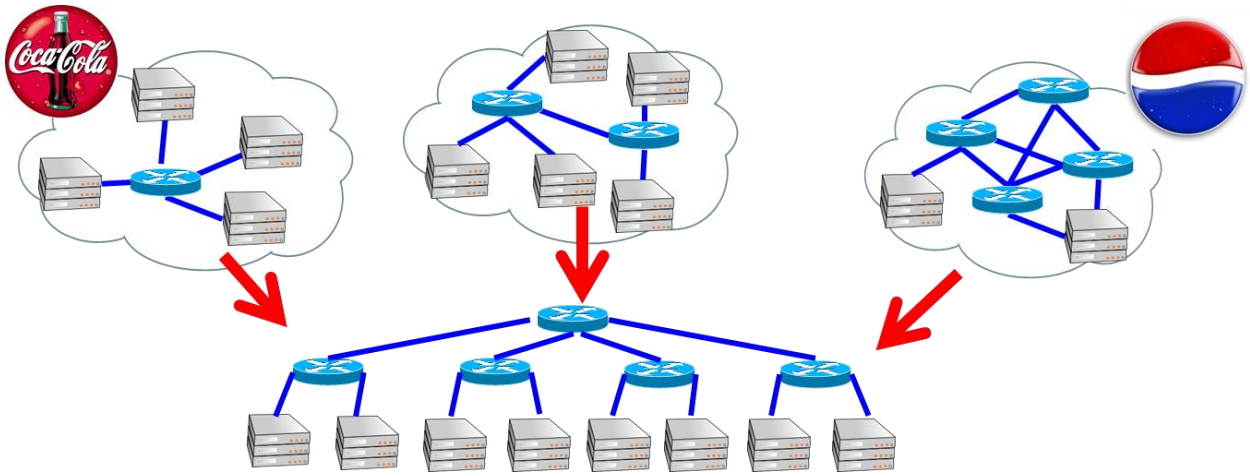

Figure 1-1: Diagram depicting interoperation between different vendors

Hosts, links, switches, and SDN/OpenFlow [1] controllers are needed to be modelled. Topologies of very large scale size (up to thousands of nodes) can be created using Mininet and they can be easily tested. It has an API and a very simple command line tool. It allows the user to easily customize, share, create and test SDN networks. It is an easily available open source software that can emulate SDN controllers and OpenFlow devices (Emulating Real Networks). It also can be used to simulate SDN networks as well as run a controller for experiments. Mininet also allows emulating the real world network scenario. A couple of SDN controllers are included within the Mininet Virtual Machine. The default controller is good, but to implement advance concepts, POX controllers are usually used. The control plane and the data plane are tightly coupled in the same devices. Therefore, in a traditional network, developing a new application and modifying the behaviour of any existing devices is a very difficult work. SDN or Software Defined networking also overcomes these problems. It does so by shifting the control logic from a device to the centralized place. This shifted control logic is known as SDN Controller or NOS (Network Operating System). The controller can have the global view of an entire network, so we can manage the functionality of any network by using SDN in an efficient manner. OpenFlow, which is a standard protocol, is used to provide the communication between a dumb device and the controller. [1]

The dumb devices and the controller are called data plane and control plane [1] respectively. The OpenFlow controller is only responsible to decide what actions are to be performed by the switches. The decision approach can be either Proactive or Reactive. In Reactive approach, whenever any packet arrives at the switch, switch doesn't know how that packet is to be handled.

So, that's why the switch sends the packet to the controller. Controller is responsible to insert a flow entry into the flow-table of the switch using OpenFlow protocol. In this approach, the main disadvantage is that the switch is totally dependent upon the controller decision. Therefore, when any switch loses its connection with the controller, it is unable to handle that packet. But in the Proactive approach, the controller pre-populates the flow entry in the flow table of each of the switches. This approach is able to overcome the limitations of the reactive approach. This can be inferred because even if the switch has lost the connection with the controller, it doesn't disrupt the traffic. The main advantage of SDN over any traditional approach is that it allows us to quickly test any real network and deploy new applications into it, also to minimize capital and reduce operating expenses. It also allows central management of each of the switches.

The traditional network architectures are very ill-suited to meet all the requirements of today's carriers, end users and enterprises. They aren't capable enough to handle MITM or Man in the Middle attacks like DDoS Attacks, ARP spoofing etc.

We need to handle coincidental wants for security, virtualization, tractability, quality and mobility in today´s networks – the conception of SDN is gaining attention as the most viable solution. The provisioning of latest services and therefore the reliable application delivery as that in a dynamic IT infrastructure may be achieved with such designs. Typically a SDN separates the information and management planes of the network and provides interfaces/APIs to provision services conjointly within the network's external systems instead of configuring device by a distributed device. SDN doesn't modify the network configuration, what it does is it simply maintains it, at a far off location (Remote Programmer).

SDN is strong, robust and provides a user flexibility to program the network in step with their wants and necessities. This project is geared towards removing a security vulnerability which is called Arp Poisoning/Spoofing from a computer code outlined Network. This is done by configuring a switch in Openflow POX (done using Python code).

1.1 OVERVIEW OF THE THESIS

Chapter 2 deals with the various dimensions of SDN and its applications. It also includes the detailed study of the various research carried out on the same subject. It also helps in getting an overview and at the same time provides a motivation to carry out the work and also gives a record of past developments on the same subject.

Chapter 3 contains the description of the different software that has been used while undertaking this work and also some of the basic commands/instructions used in the software are given also giving the result of using those commands.

Chapter 4 deals with the code and algorithm that has been used in the whole of the project work. The first code is about creating custom topologies and second is that of ARP detection.

Chapter 5 deals with the simulation results and their analysis. Snapshots were taken of the system in which all the work was carried out.

Chapter 6 deals with the conclusion of results and the future scope of the project. It also contains discussion/conclusion which can be inferred from the simulations. It also provides an insight to the myths and realities about SDN and also suggests further work that can be undertaken.

CHAPTER 2

Literature Review

"In today's world, where there is existence of large data centres for storing and manipulating data from a large number of databases, SDN becomes a very easy way to access data across network architectures through the use of OpenFlow protocol which enables user from one network architecture to interact with other networks."

2.1 Mininet

Mininet [2] is an emulator and is used for deploying very large networks on any Virtual Machine or on limited resources of any simple single computer.

Mininet has enabled researchers to research the areas of in OpenFlow and Software Defined Networking (SDN) [2]. Mininet emulator allows us to run unmodified version of code interactively on a virtual hardware in a simple PC. It also provides us convenience and realism at a really very low cost. The alternatives to Mininet are hardware test beds which are very fast, also accurate but very expensive and are shared. Another option can be to use simulators which are very cheap but are sometimes slow and also require code modifications. Mininet also offers accuracy, ease of use, scalability and performance.

2.2 Network Architecture

Virtual Computing has indeed dramatically increased importance of network infrastructures. In today's age, after we have had the era of mainframe, Internet computing and client/server, virtualized applications are now hosted in the public and private cloud – ultimately enabled accessibility to mobile users and devices from any place. Networks have gradually become the critical component in such an infrastructure.

All networks are built using routers, switches and other devices in distributed fashion to scale the size and also provide reliability. In this kind of distributed environment, it has become much more complex to provide any new end-to-end service and application in a cost effective and seamless manner.

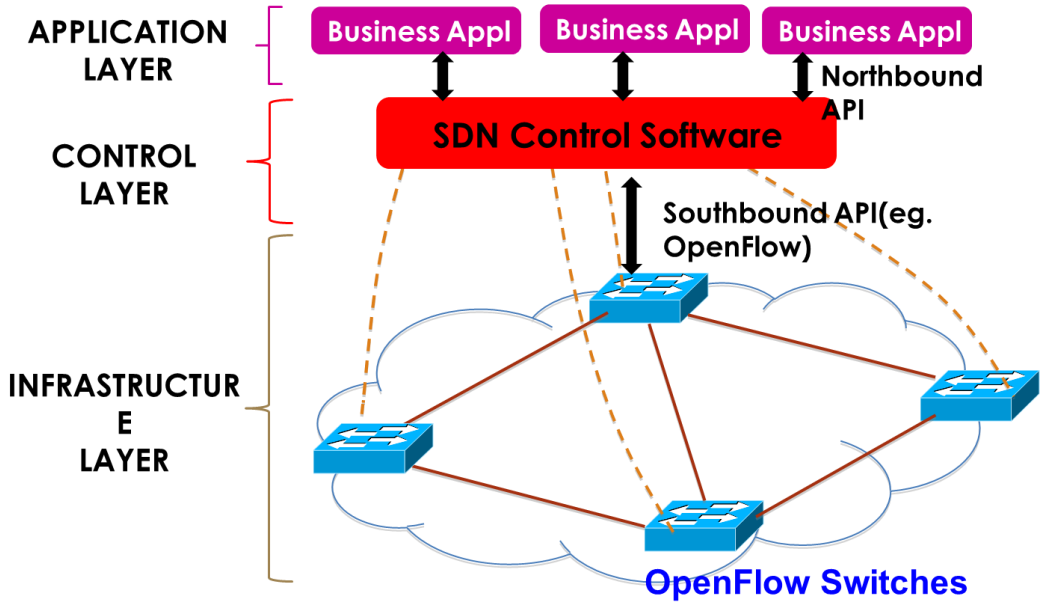

Figure 2-1: Traditional Network Infrastructure

The networking system presently used across the world are categorized unremarkably in two:

- **LAN (Local area Network):** Network in an institute or a building.

- **WAN (Wide area Network):** Network in a very larger geographical region. Ex: web is taken into account to be the most important WAN spanning the entire Earth.

Every Network has 2 components:

- **Control Plane:** a part of the router design that's involved with drawing the network map, or the data within the routing tables that defines what to try to do with the incoming packets.

- **Data Plane:** a part of the router design that decides what is to be done with packets coming on an inward interface. Most ordinarily, it refers to a table during which the router usually looks up the destination address of the incoming packet and retrieves the data necessary to see the trail from the receiving part, through the interior forwarding material of the router, and to the correct outgoing interface(s). It's conjointly referred to as Forwarding Plane. [3]

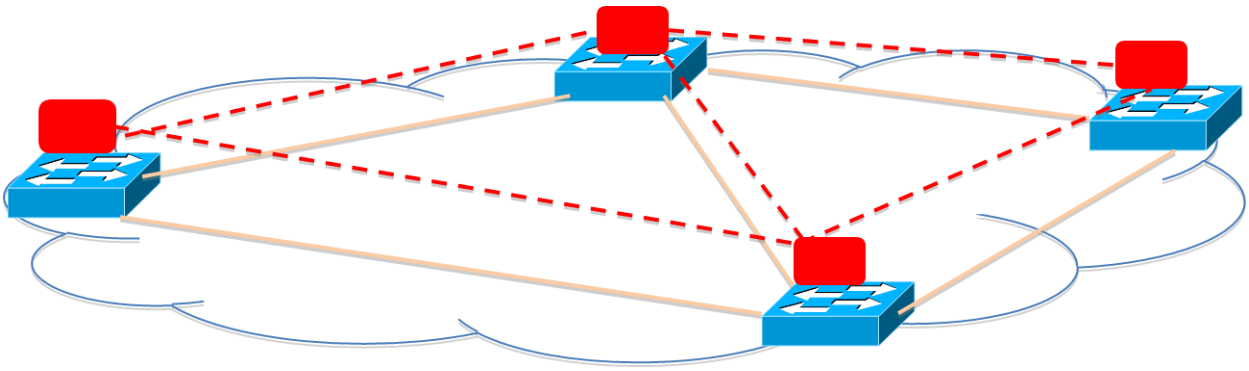

Figure 2-2: Control Plane and Data Plane

For example in Figure one we've got an easy three router topology that's R1–R2–R3, and R1 and R3 square measure running OSPF with one another. From R1 and R3's perspective, these packets are a part of the management plane, as a result of their regionally originated/destined, and want to be method switched so as to appear into the packet details. But from the R2's perspective, it sees these packets would be in its data plane, this is so because traffic is neither originated from nor destined to that.

[1]Processing Plane	Where it runs	How fast these processes run	Types of processes performed
Control Plane	Switch CPU	In the order of thousands of packets per second	Routing protocols (i.e. OSPF, IS-IS, BGP), Spanning Tree, SYSLOG,AAA(Authentication Authorization Accounting), NDE(Netflow Data Export), CLI(Command Line interface), SNMP
Data Plane	Dedicated Hardware ASIC's	Millions or billions of packets per second	Layer 2 switching, Layer 3 (IPv4 | IPv6) switching, MPLS forwarding, VRF Forwarding, QOS (Quality of Service) Marking, Classification, Policing, Netflow flow collection, Security Access Control Lists

Figure 2-3: Properties of Control and Data Plane

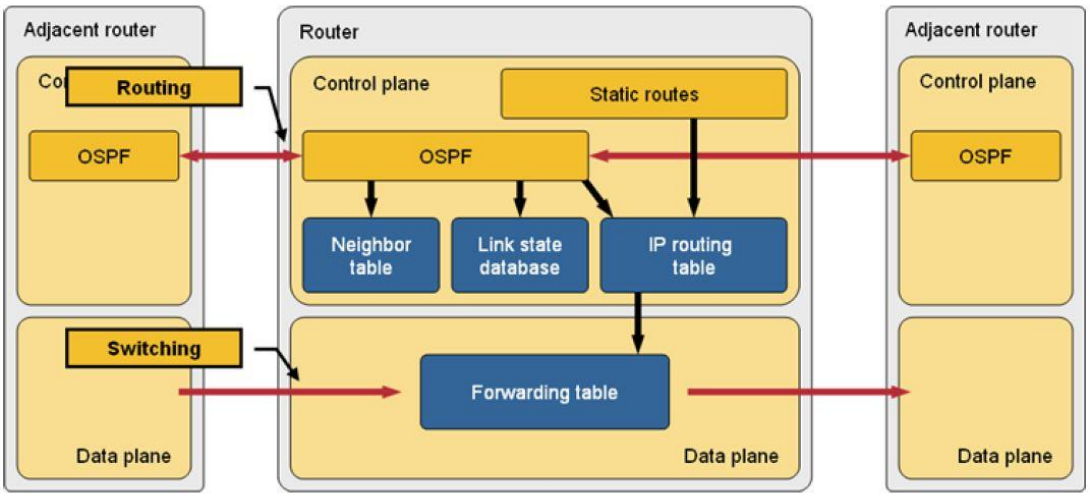

Figure 2-4: Interconnection between 3 routers

2.3 Limitations of Traditional Networks

- The current networks are very complex given that there are a mix of switches and routers that work together and the addresses being pre-defined. Hence all the address are stored in a database so for large networks, it becomes more and more complex owing to the higher number of hosts. This system is also static as we cannot redefine the stored values of a router once it is defined. So this complexity and the resulting static nature is a big disadvantage.

- Another major disadvantage is that each vendor has its own network policy for its own network defined for all the switches, routers and hosts in that network. Hence it becomes nearly impossible to smoothly transfer data from one system of network to another. So interoperability is a major issue. [3]

- As all the router mechanisms are pre-defined, hence it is impossible to add a new router as that would mean that we have to change the definition of all routers and hence the network becomes larger and more complex.

- Each supplier has its own definition and protocols of controlling the network. So a user in one network cannot work on the other network without learning the mechanism of the new network.

Control and data Plane Separation

- In SDN, control and data planes are separated. Within the SDN paradigm, not all the process happens within the same device. It aggregates/consolidates the control planes.

- There is just one controller dominating the complete network and all the network parts like router, switch etc. act like dumb devices and are operated via the controller.

2.4 POX

POX is said to be an open source development platform for Python-based software-defined-networking (SDN) management applications, like OpenFlow SDN controllers. It's a high-level SDN API as well as a queriable topology graph and support for virtualization. [1]

- "Pythonic" OpenFlow interface.
- Reusable sample elements for path choice, topology discovery.
- "Runs anywhere" – will bundle with install-free PyPy runtime for straightforward readying.
- Specifically targets UNIX operating system, Windows and MAC OS.
- Supports the same interface and visual image tools as Nox.
- Performs well compared to Nox (a sister project) applications written in Python.

2.5 Advantages of SDN

- Centralized management of multi-vendor environments: The OpenFlow-enabled network devices or components like routers, ports and switches from different vendors will all be automated, arranged managed to be deployed across the network instead of manual intervention.

- Reduced complexity of network through automation: The network instability and errors associated with it can be ridden off by use of SDN tools that can automize and manage the tasks which are usually performed manually.

- Higher rate of innovation: A virtual network can be abstracted from individual network services by programming and reprogramming it in real time. This leads to higher business innovation for the IT operators.

- Increased network dependableness and security: The policy statements and configuration parameters are usually translated into the network infrastructure via Openflow. The associate open flow based SDN reduces the probability of network failure owing to the policies and configuration settings.

- More granular network management [3]: The SDN controllers provide transparent control over network and thus engineering the traffic, management, providing Quality of service are taken care of over all the infrastructures. The advantage is less errors and consistent configuration governed by systematic enforcement of rules.

- Better user experience: The Open Flow based SDN takes care of dynamic user requirements. One such example is a video application where the resolution need not be set by the user without knowledge of what the network can support but the application itself can find information measures that can be accessed by the network and regulate it accordingly.

2.6 OpenFlow

- The ONF [4] or what we call Open Networking Foundation is an organization led by the users and is completely dedicated to the adopting, materializing, developing and advertising of software-defined networking (SDN). It is managed by the OpenFlow standard.

- ONF has defined OpenFlow as the very 1st standard communication interface or GUI defined between the controls and forwarding layers of the SDN network/architecture.

- The major advantage of using OpenFlow is that it allows direct access and complete manipulation of the network device's forwarding plane such as that of the switches & routers, both virtual and physical.

- OpenFlow, like other protocols, is essentially required to move the network controls out of the proprietary network switch and directly into the control software - that's open source and locally managed.

- OpenFlow also essentially allows network packet's path through the network of the switches which are to be determined by software which is running on multiple routers. The separation, such as this, of the control from the forwarding plane allows users for

better traffic management and is rather feasible using routing protocols and Access Control Lists (ACLs).

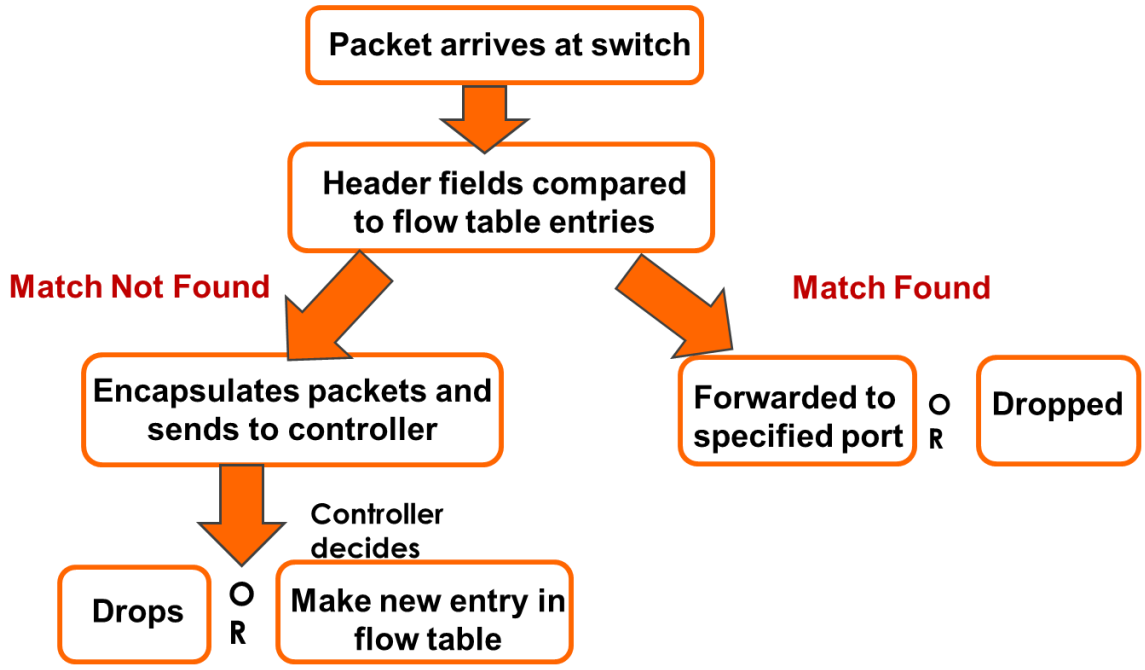

Figure 2-5: Openflow Mechanism

2.7 Use of SDN in Large Data Centres

- In today's world, where there is existence of large data centers for storing and manipulating data from a large number of databases, SDN becomes a very easy way to access data across network architectures through the use of OpenFlow protocol which enables user from one network architecture to interact with other networks.

- SDN uniquely helps users to overcome this problem. This is done so by creating isolated networks which are logically connected. [2]Then SDN uses the slicing technique to partition them and interact with each other.

- Using SDN here means to basically to abstract the management section of control plane of each of the member devices of the network into a central scheme consisting of a controller, this can be done using the OpenFlow protocol. This in turn helps us to isolate networks and helps it to grow within themselves, at the same time enabling them to interact with each other.

Figure 2-6: Different Vendors for Network Protocol

2.8 SDN Controllers

A Software defined Networking (SDN) controller can be said to be an application that holds the ability of control flow management which enables us to implement intelligent networks. The controller can be based on various protocols, one such protocol is OpenFlow, which is the most commonly used, which can allow the existing servers to direct the path of the packets.

It is the link between the devices in the network and the application layer of the network and thus forms the core of the network architecture. All the communications that takes place between the devices and the network applications, goes through the controller. At the same time, the controller with the help of protocol can find the optimal path through the network to control the traffic and also configure the network devices.

So, in short the controller can be considered as an Operating system of the networks. It enables to separate the control plane from the hardware devices and instead program it on software. This greatly helps to make the system automated and also easier to administer and also integrate all the applications.

Hence it can be seen that the controller is a strategic controlling point in the network and acts as its “brain” as it relays the vital information between the switches/routers from the layer below to the application layer which is above. The SDN controllers existing today are basically plug-in devices i.e various modules can be inserted into it to perform different tasks [5]. Tasks can vary from collecting statistics of the network to running certain algorithms to analyse or induct fresh rules in the network paradigm.

The figure given below shows some of the vendors who provide SDN controllers. The most commonly used protocol is OpenFlow but others are being developed recently. The protocol being used is pivotal as it can define the architecture of the network and the decision making of the controller.

Figure 2-7: Vendors producing SDN Controllers

2.9 Spoofing/Poisoning of ARP

ARP (Address Resolution Protocol) is a part of what we call the Man in the Middle Attack (MITM) [5] in which the attack involves modification of network traffic, or intercepting of frames of data or stopping the network completely. The attack is usually is seen as a beginning of a series of such attacks in which the attacker tries to hijack the session or stop the services.

The IP address gets converted to MAC address while an IP datagram transmission. An ARP request is generated when other host's IP address in known without the knowledge of the MAC address. In this case a broadcast packet is sent out to Local network.

ARP is a stateless protocol where the ARP reply contains the MAC address of the required IP. ARP replies are usually cached and the entries which have not expired will be over written with the arrival of new packet. The identity of the peer from where ARP is generated is tough to determine and this causes ARP spoofing. ARP spoofing exploits this vulnerability by sending spoofed ARP messages via LAN.

The aim is to associate the attacker's MAC address with the IP address of target host. The attacker could modify the data (corruption in data), launch a denial-of-service attack (DoS) or forward it to the actual default gateway, thus causing some or all packets on the network to be dropped. [5]

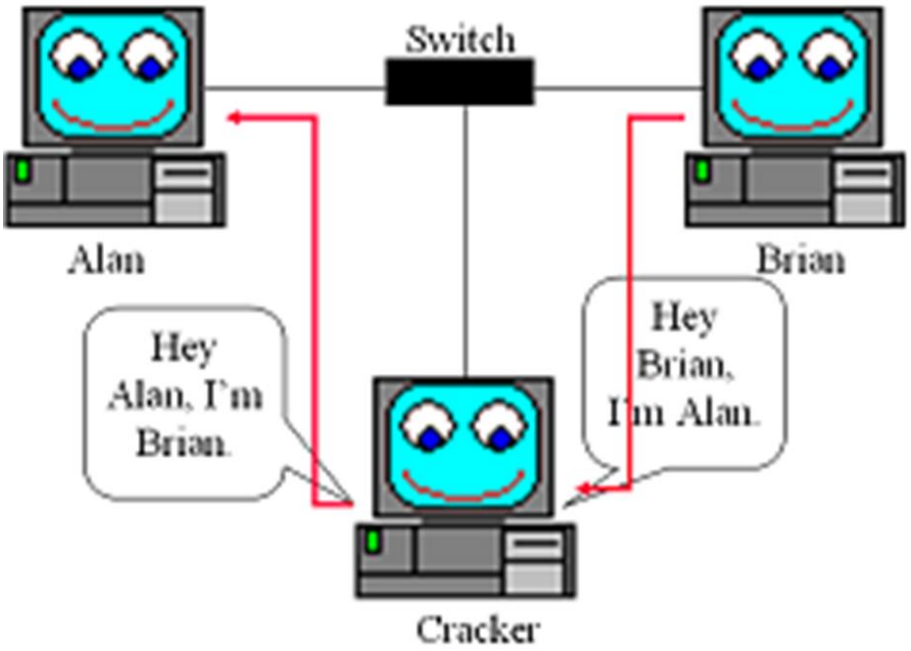

Figure 2-8: ARP Spoofing

2.10 Network Virtualization

FlowN [12] and FlowVisor are two addressing mechanisms that has been proposed for virtualization of the network so as to enable different networks to interact with each other. Both the methods involve the slicing up of the address space (as shown in fig) and allow each stakeholder (tenant) to control its own network. Each tenant is illusioned to be having its own address space, its own controller and topology. They use the technology of database management to efficiently manipulate and store all the mappings. This is done by a FlowN controller rather than individual controllers of the network. [6]

The FlowN and FlowVisor methods work on the same method of splicing up of memory but vary in the way that in FlowVisor the address space of one tenant is visible to the other but in FlowN it is not visible. So FlowN is better in terms of security and through simulations it can be shown that it is better in speed as well.

Network Virtualization is still a distant dream owing to the scalability issue when a large number of tenants are present and hence making the system slower. Still work is being undertaken to find a suitable way to realize virtualization and it has become the primary motive of most of the vendors.

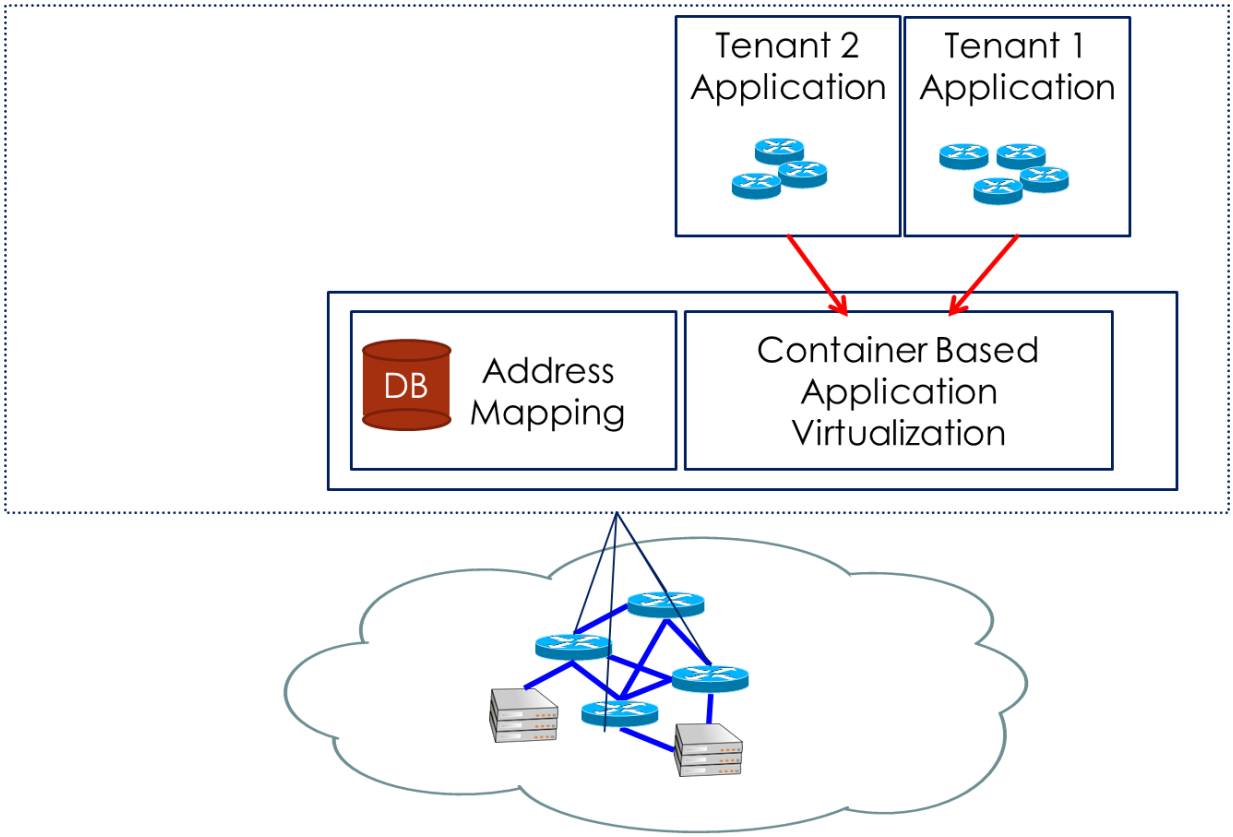

Figure 2-9: FlowN Infrastructure

CHAPTER 3

Software Used

"The VirtualBox version used for the project was freely available in the official Mininet website and came with the virtual OS containing wireshark, mininet terminals, Xming servers and python compilation tool. These were all pre-installed in the virtual OS."

3.1 Oracle VM VirtualBox

This is a very powerful oracle product used for virtualization and is full of features, delivers high speed performance and most importantly is a free source i.e. is available for free. The VirtualBox version used for the project was freely available in the official Mininet website and came with the virtual OS containing wireshark, mininet terminals, Xming servers and python compilation tool. These were all pre-installed in the virtual OS.

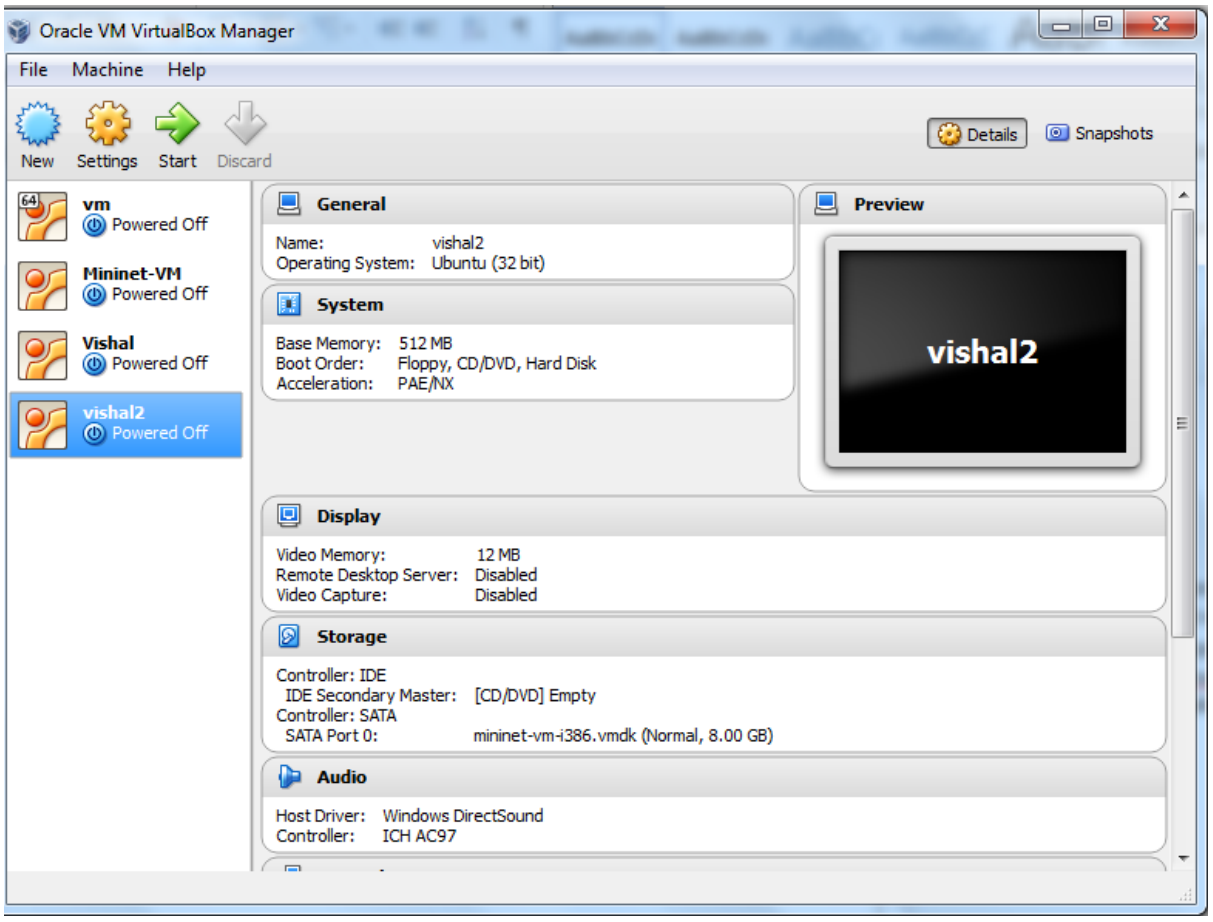

Figure 3-1: Oracle VirtualBox

3.2 Ubuntu Operating System

The VM image which was available for free in the website had an Ubuntu Operating System. The version of the OS was 14.04 for 32-bit version. All the experiments were carried out in this OS in the VM itself without installing it directly into the host computer.

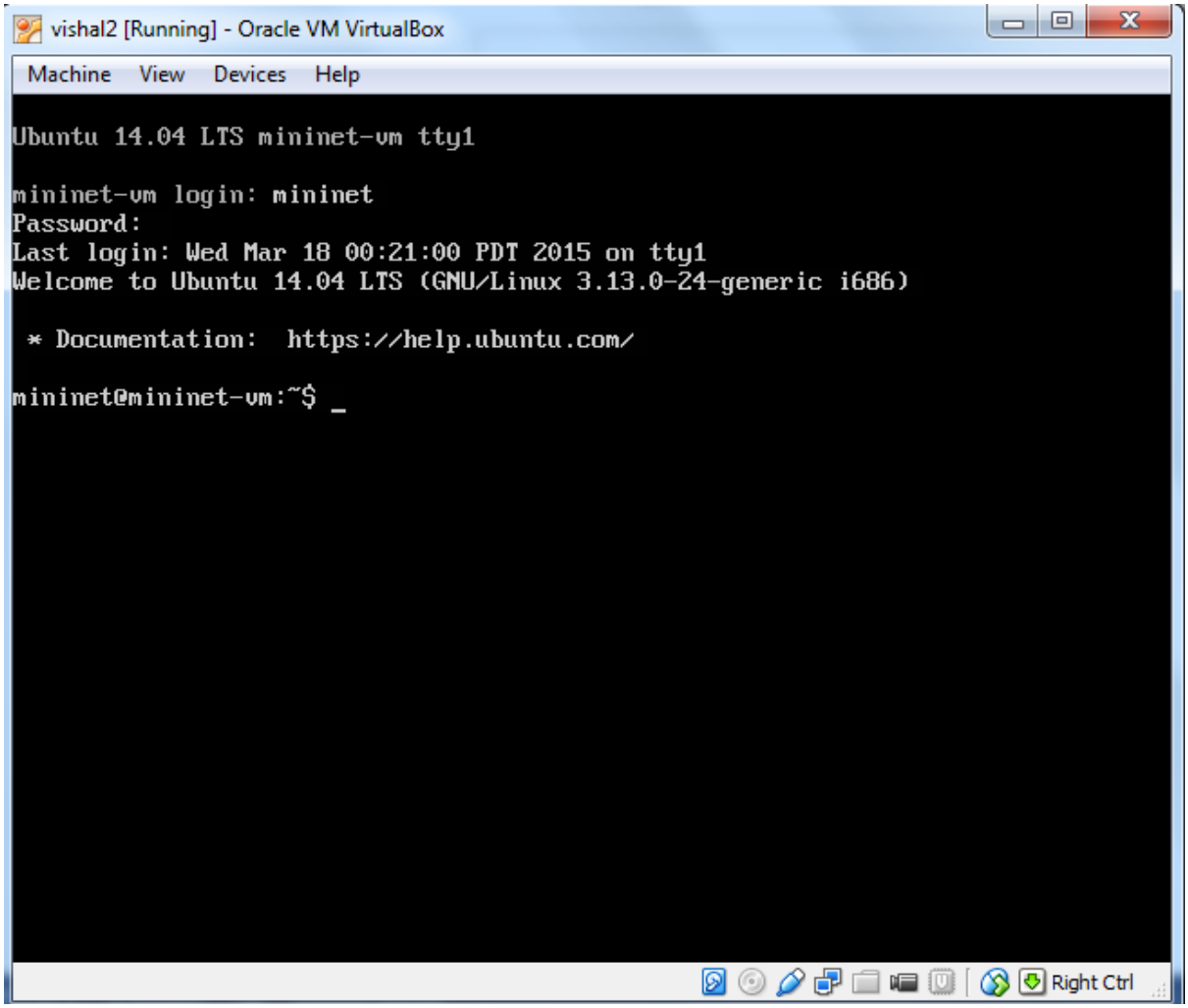

Figure 3-2: Ubuntu in Oracle VM

3.3 Putty

A SSH connection can be created between the OS(windows or linux) and the mininet so that command lines can be executed in the mininet Virtual Machine.

3.4 Xming Server

The purpose of Xming is to provide X11 server for minimal yet functional use in the windows environment. Thus the users can execute on sorts of graphic application in workstation which is remote and which will not need a high amount of space on hard disk.

3.5 Wireshark

Wireshark in mininet is used for the purpose of analysing the traffic and capturing the packets that are running in the network.

It can be run using the command “sudo wireshark &”

Then we choose the loopback interface to capture any data and a filter can be setup by typing the command “of”. This is a filter for OpenFlow traffic.

Now we can open the OpenFlow reference controller. This is done by typing “controller ptcp:” in the SSH terminal. This command thus starts a controller which can act like a learning switch that too without the installation of any flow entry and variety of message can be seen.

Message	Type	Description
Hello	Controller>Switch	following the TCP handshake, the controller sends its version number to the switch
Hello	Switch>Controller	the switch replies with its supported version number
Features Request	Controller>Switch	the controller asks to see which ports are available
Set Config	Controller>Switch	Controller asks the switch to send flow expirations
Feature Request	Switch>Controller	the switch replies with a list of ports, port speeds, and supported tables and actions.
Port Status	Switch>Controller	enables the switch to inform that controller of changes to port speeds or connectivity. Ignore this one, it appears to be a bug

Figure 3-3: Wireshark Commands

3.6 Mininet

Basically we need to emulate the network using mininet. Some of the common commands and their functions are given in the figure.

Mininet>nodes

```
available nodes are:
c0 h1 h2 s1
```

Mininet>dump

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3633>
<Host h2: h2-eth0:10.0.0.2 pid=3634>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=3637>
<OVSController c0: 127.0.0.1:6633 pid=3625>
```

Mininet>net

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

Let see host 1 ping host2 in default topology

Mininet> h1 ping h2

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_req=1 ttl=64 time=2.18 ms
64 bytes from 10.0.0.2: icmp_req=2 ttl=64 time=0.362 ms
64 bytes from 10.0.0.2: icmp_req=3 ttl=64 time=0.053 ms
64 bytes from 10.0.0.2: icmp_req=4 ttl=64 time=0.083 ms
64 bytes from 10.0.0.2: icmp_req=5 ttl=64 time=0.088 ms
^C
--- 10.0.0.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 0.053/0.553/2.180/0.821 ms
mininet>
```

Figure 3-4: Mininet Commands 1

Pingall command let you ping from every host to every host

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
mininet>
```

To see TCP and UDP bandwidth between hosts the command is iperf

Mininet>ipref

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
waiting for iperf to start up...*** Results: ['9.12 Gbits/sec', '9.13 Gbits/sec'
]
mininet>
```

Figure 3-5: Mininet Commands 2

All the codes were written using Python programming language. The compiler was pre-installed in the Mininet VM. These topologies are run in Mininet by using commands. Python API use its own classes, functions, methods and variables to create custom type topologies.

CHAPTER 4

Algorithms and Codes

"A network consisting of a switch and three hosts was created and the code named as antiArpPoisoning which is written in python was run."

4.1 Code for creating a custom Topology

This code creates a network of 4 hosts and 4 switches. Complex networks can be created by using this code prototype. [7]

```
from mininet.topo import Topo
from mininet.node import OVSSwitch, Controller, RemoteController
c0 = RemoteController( 'c1', ip='192.168.81.132')
c1 = RemoteController( 'c2', ip='192.168.81.130')
cmap = { 's1': c0, 's2': c0, 's3': c1,'s4':c1 }
class MultiSwitch( OVSSwitch ):
   def start( self, controllers ):
      return OVSSwitch.start( self, [ cmap[ self.name ] ] )
class OnosTopo( Topo ):
   "Simple topology example."
   def __init__( self ):
      "Create custom topo."

 Topo.__init__( self )
      h1 =  [ self.addHost( 'h1')]
      h2 =  [ self.addHost( 'h2')]
h3 =  [ self.addHost( 'h3')]
  h4 =  [ self.addHost( 'h4')]
   s1 = [ self.addSwitch( 's1', dpid="0000000000000201")]
   s2 = [ self.addSwitch( 's2', dpid="0000000000000202")]
   s3 = [ self.addSwitch( 's3', dpid="0000000000000203")]
   s4 = [ self.addSwitch( 's4', dpid="0000000000000204")]
   self.addLink('s1','h1')
   self.addLink('s2','h2')
   self.addLink('s3','h3')
   self.addLink('s4','h4')
   self.addLink('s1','s2')
   self.addLink('s3','s4')
topos = { ‘mytopo': ( lambda: myTopo() ) }
```

4.2 Code to Detect and Remove ARP Spoofing

The algorithm is defined as below:

- A network consisting of a switch and three hosts was created and the code named as antiArpPoisoning which is written in python was run.

- IP address was assigned to all the 3 hosts which was named as “10.0.0.2”, “10.0.0.3”, “10.0.0.4” and 10.0.0.1 was declared as the default gateway.
- A network flow was then started among the 3 hosts by using the ping command.

- The network is spoofed by running the command “*h1 arpspoof –i h1-eth0 –t 10.0.0.2 10.0.0.3*” and thus the 1st host poisons the network between the other two.

- The spoof is detected by the controller which in turn stops the flow of data until the network gets timed out.

```
from pox.core import core
import pox.openflow.libopenflow_01 as of
from pox.lib.recoco import Timer
from pox.lib.util import dpidToStr
from pox.lib.addresses import IPAddr, EthAddr

log = core.getLogger()
TimerQuantumDuration = 10

class AntiARPPoisonSwitch (object):
def __init__ (self, connection):
self.connection = connection
connection.addListeners(self)
self.mac_to_port = {}
self.ip_to_mac = {}
self.ip_to_mac_time_track = {}
Timer(TimerQuantumDuration, self.Clean_Tables, recurring = True)
```

```
def send_packet (self, buffer_id, raw_data, out_port, in_port):
msg = of.ofp_packet_out()
msg.in_port = in_port
if buffer_id != -1 and buffer_id is not None:
msg.buffer_id = buffer_id
else:
if raw_data is None:
return
msg.data = raw_data
action = of.ofp_action_output(port = out_port)
msg.actions.append(action)
self.connection.send(msg)

def arp_spoof_detected(self, packet, packet_in):
ipsrc=self.int_of_ip(packet, packet_in)
arp = packet.find('arp') if arp is not None:
macsrc=str(arp.hwsrc)

ip = packet.find('ipv4')
if ip is not None:
macsrc=str(packet.src)
if ipsrc in self.ip_to_mac:
if self.ip_to_mac[ipsrc] != macsrc:
return True
self.ip_to_mac[ipsrc] = macsrc
return False

def check_arp_spoof(self, packet, packet_in):
if self.arp_spoof_detected(packet, packet_in):
```

```
arp = packet.find('arp')

if arp is not None:

tempipstring=str(self.ip_to_mac[arp.protosrc.toUnsignedN()])

print "Dup IP to MAC!!!"+ str(arp.protosrc)+"already taken by "+ tempipstring + ". " +
str(arp.hwsrc) + " may be spoofing!"


def int_of_ip (self, packet, packet_in):

arp = packet.find('arp')

if arp is not None:

ipsrc=arp.protosrc.toUnsignedN()

ip = packet.find('ipv4')

if ip is not None:

ipsrc=ip.srcip.toUnsignedN()

return ipsrc


def act_like_switch (self, packet, packet_in):

self.mac_to_port[str(packet.src)] = packet_in.in_port

self.check_arp_spoof(packet, packet_in)

if self.arp_spoof_detected(packet, packet_in):

log.debug("Installing drop flow...")

msg = of.ofp_flow_mod()

msg.match=of.ofp_match(dl_src=packet.src)

msg.idle_timeout = TimerQuantumDuration * 2

msg.actions.append(of.ofp_action_output(port=100))

self.connection.send(msg)

elif (str(packet.dst) in self.mac_to_port):

log.debug('MAC '+str(packet.src)+' on port '+str(packet_in.in_port))
self.send_packet(packet_in.buffer_id,packet_in.data, self.mac_to_port str(packet.dst)],
packet_in.in_port)

log.debug("Installing flow...")
```

```
msg = of.ofp_flow_mod(cookie=self.int_of_ip(packet,
packet_in),flags=of.OFPFF_SEND_FLOW_REM)

msg.idle_timeout = TimerQuantumDuration

msg.actions.append(of.ofp_action_output (port = self.mac_to_port[str(packet.dst)]))


self.connection.send(msg)

else:

self.send_packet(packet_in.buffer_id, packet_in.data,of.OFPP_FLOOD, packet_in.in_port)


def _handle_PacketIn (self, event):

packet = event.parsed

if not packet.parsed:

log.warning("Ignoring incomplete packet")

return

packet_in = event.ofp self.act_like_switch(packet, packet_in)


def _handle_FlowRemoved (self, event):

log.debug("Flow removed on switch %s", dpidToStr(event.dpid))

if event.ofp.cookie in self.ip_to_mac:

del self.ip_to_mac[event.ofp.cookie]


def Clean_Tables(self):

print "Show Table" print

self.ip_to_mac


launch():

def start_switch (event):

log.debug("Controlling %s" % (event.connection,))

AntiARPPoisonSwitch(event.connection)

core.openflow.addListenerByName("ConnectionUp", start_switch)
```

CHAPTER 5

Simulations and Results

"Wireshark is used in Mininet to monitor the flow of packets and at the same time calculate essential data such as bandwidth, delay, latency and time period of data. This data can further be used to optimize the network and compare the results of various algorithms."

5.1 Creating a Network in Mininet

A simple network of a certain number of hosts and switches can be created by using the single command.

```
sudo mn --switch ovsk --controller ref --topo tree,depth=2,fanout=8 --test
pingall
```

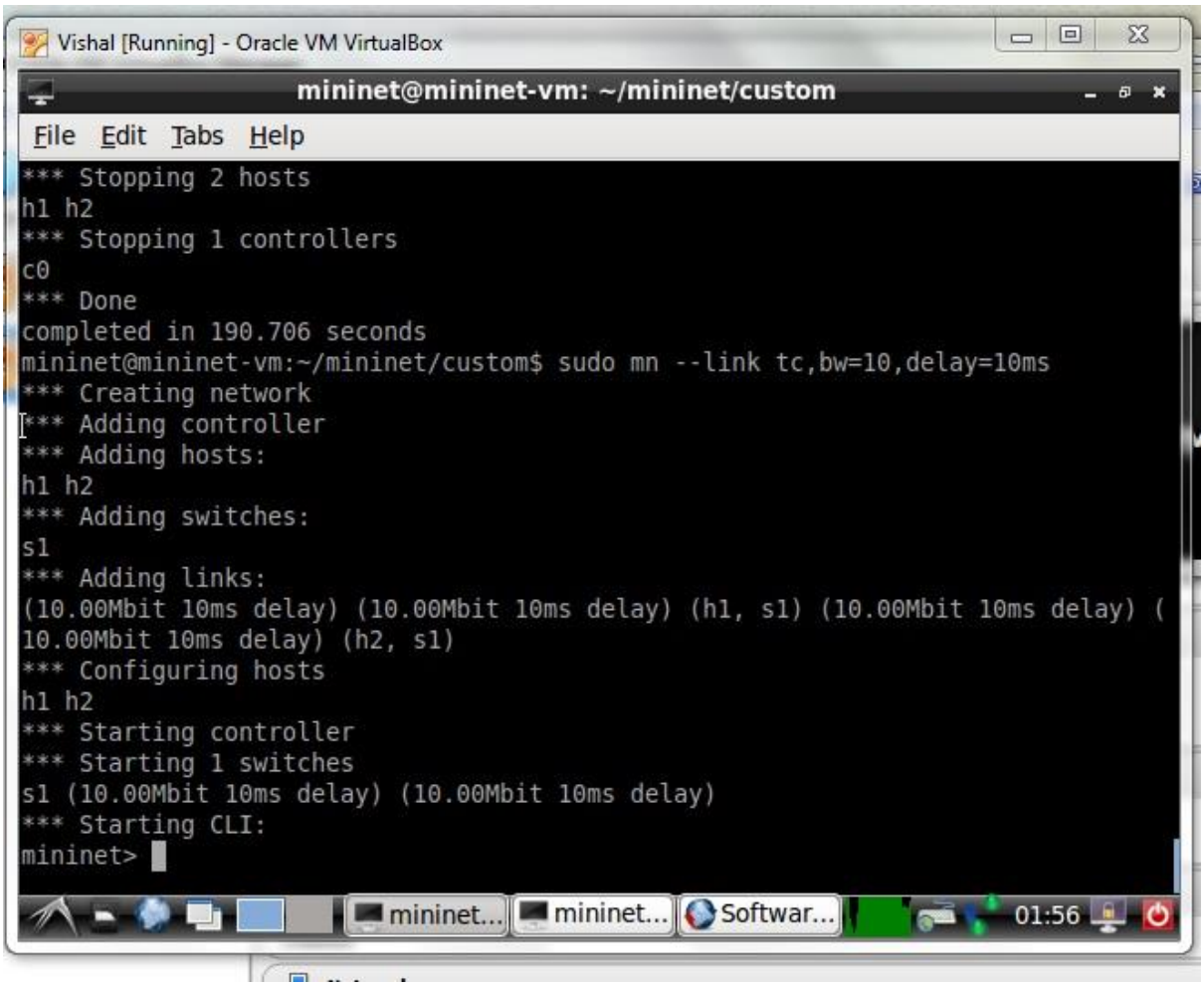

Figure 5-1: Creating a network in Mininet

5.2 Running Custom Topologies

Once a custom topology has been written using Python coding and saved in the VM's memory. It can be run using the mininet terminal by running the .py file.

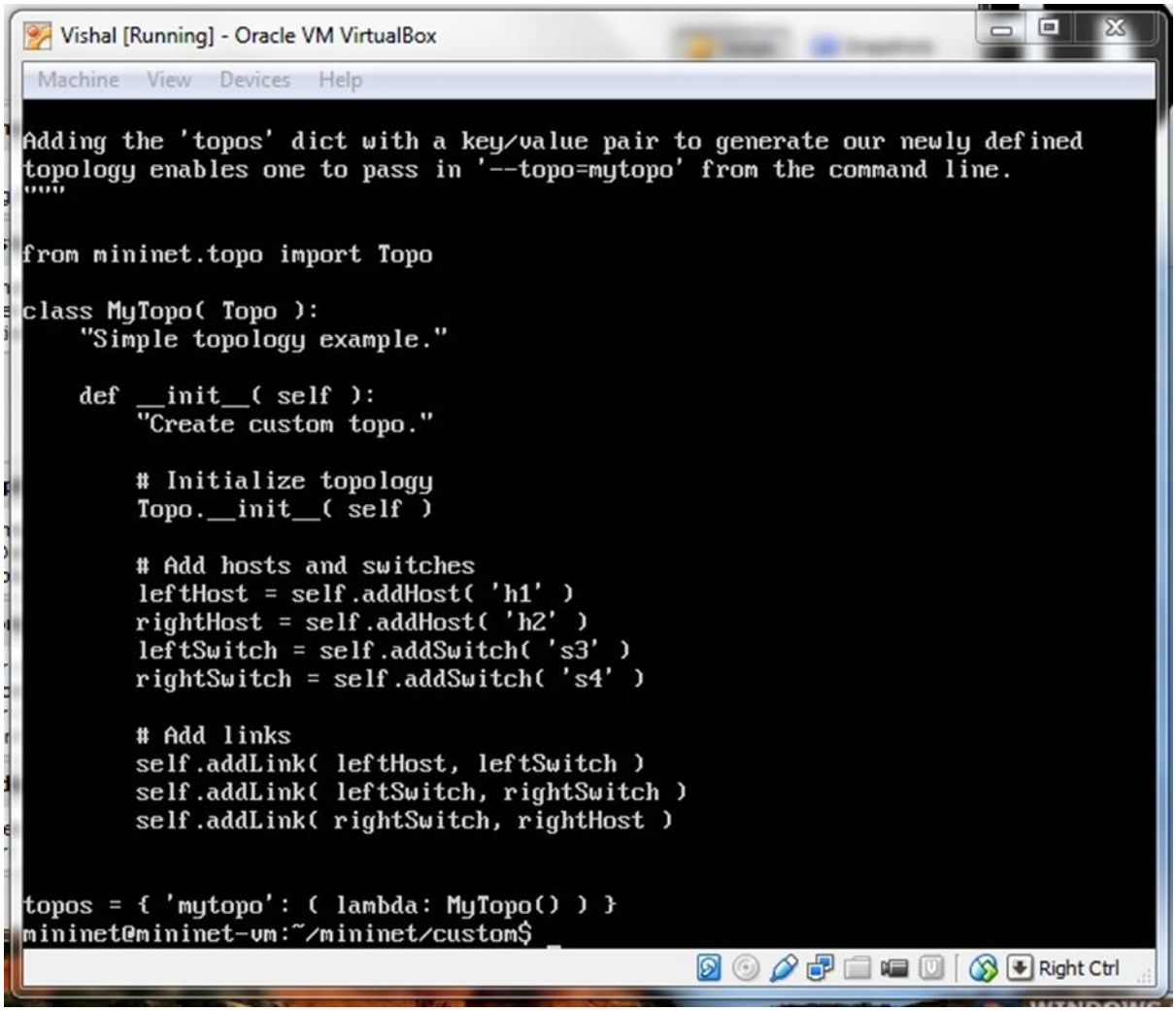

Figure 5-2: Custom Topology Python file

5.3 Network Interaction in Mininet

The CLI in the Mininet allows the user to manage and hence control the entirety of the virtual network. This can be done by using the ping command.

```
mininet> h2 ping h3
```

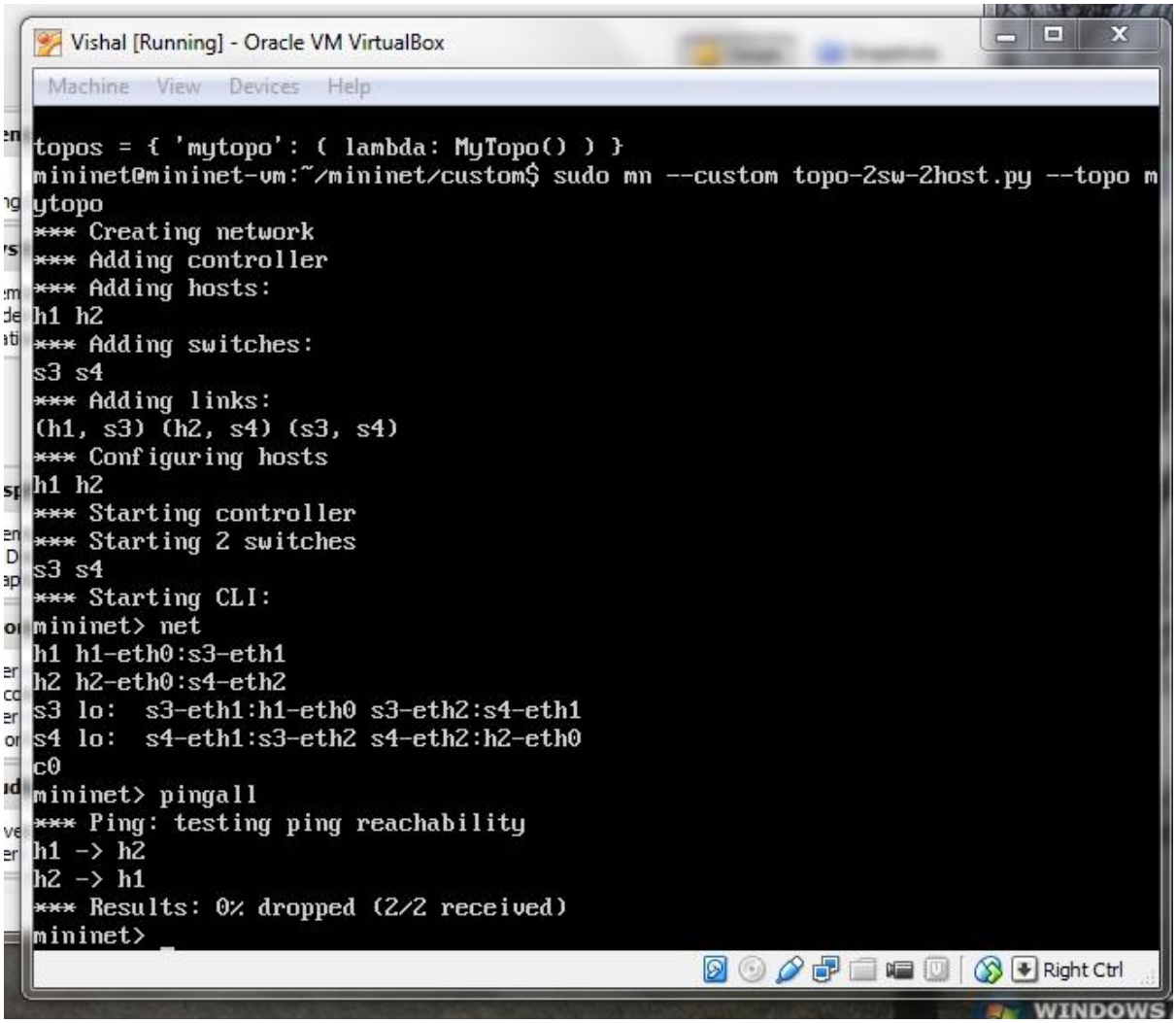

Figure 5-3: Interacting in VM

5.4 ARP Detection and Prevention

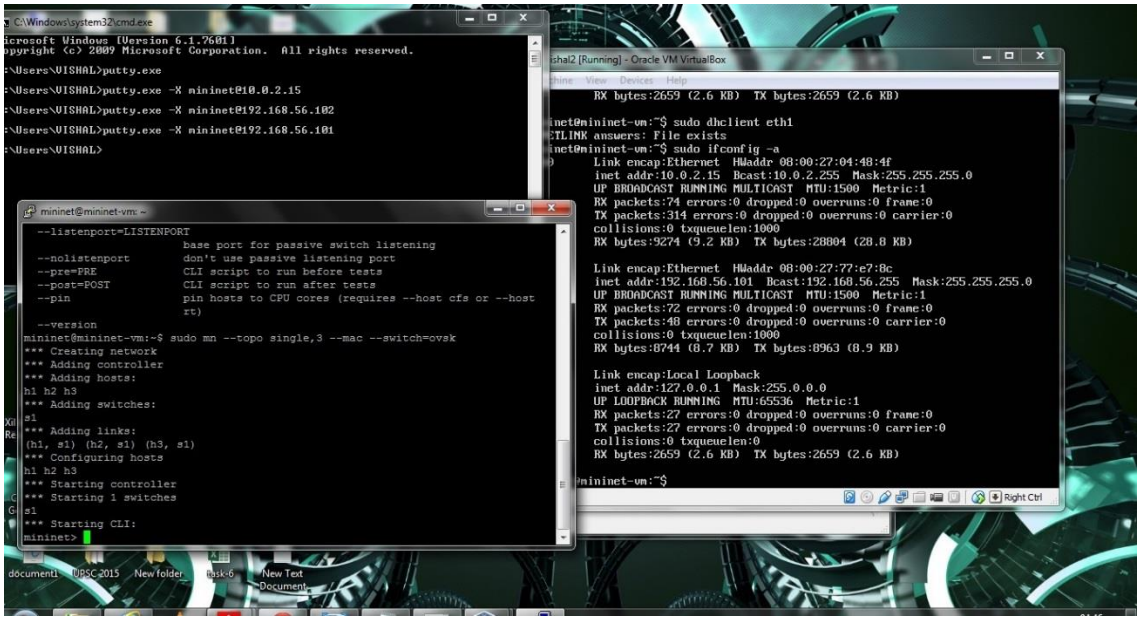

Figure 5-4: Creating network for ARP Spoof Detection

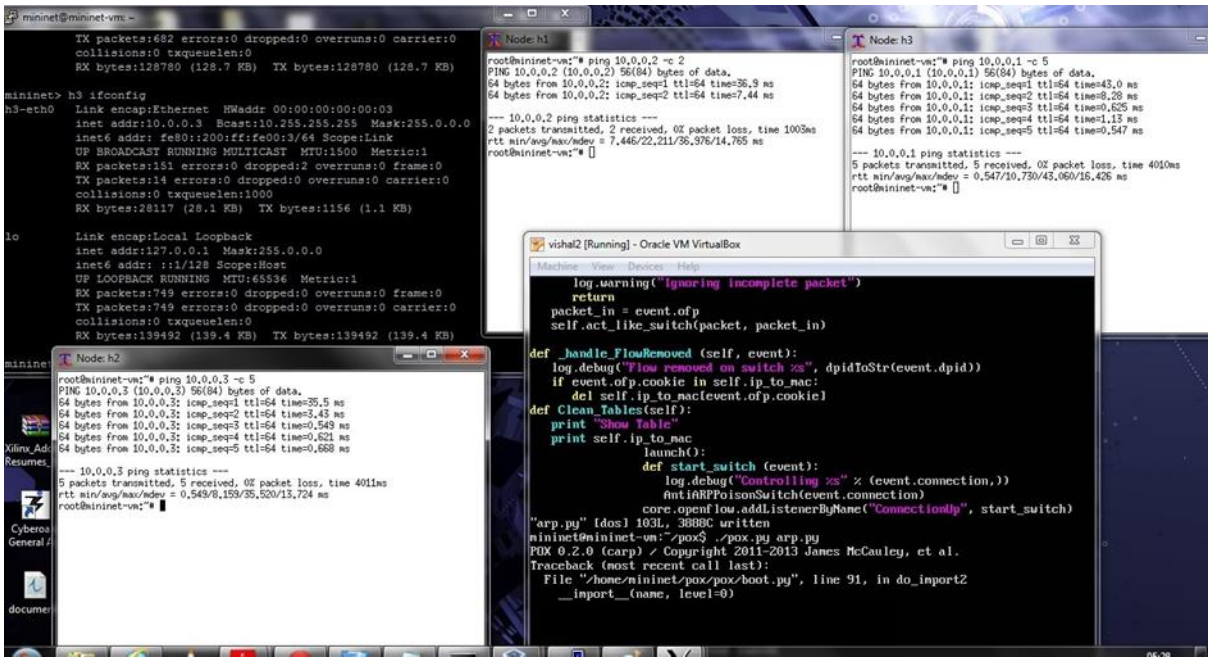

Figure 5-5: Poisoning the network

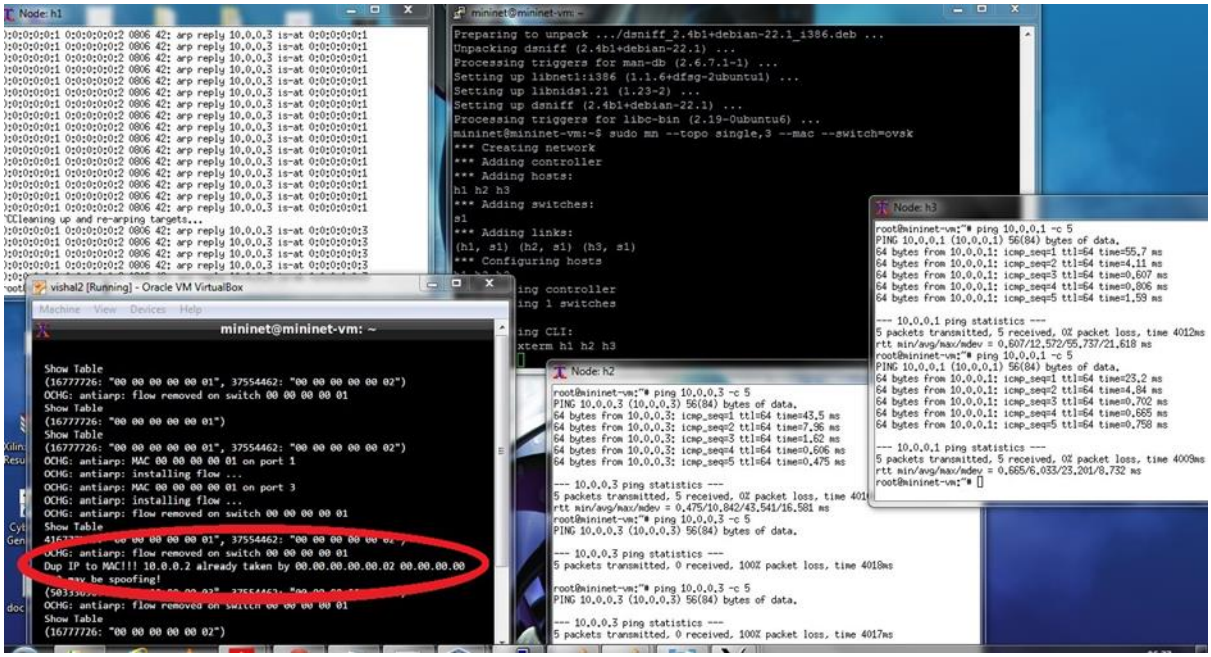

Figure 5-6: Detecting the ARP Spoof

5.5 Network Monitoring Using Wireshark

Wireshark is used in Mininet to monitor the flow of packets and at the same time calculate essential data such as bandwidth, delay, latency and time period of data. This data can further be used to optimize the network and compare the results of various algorithms.

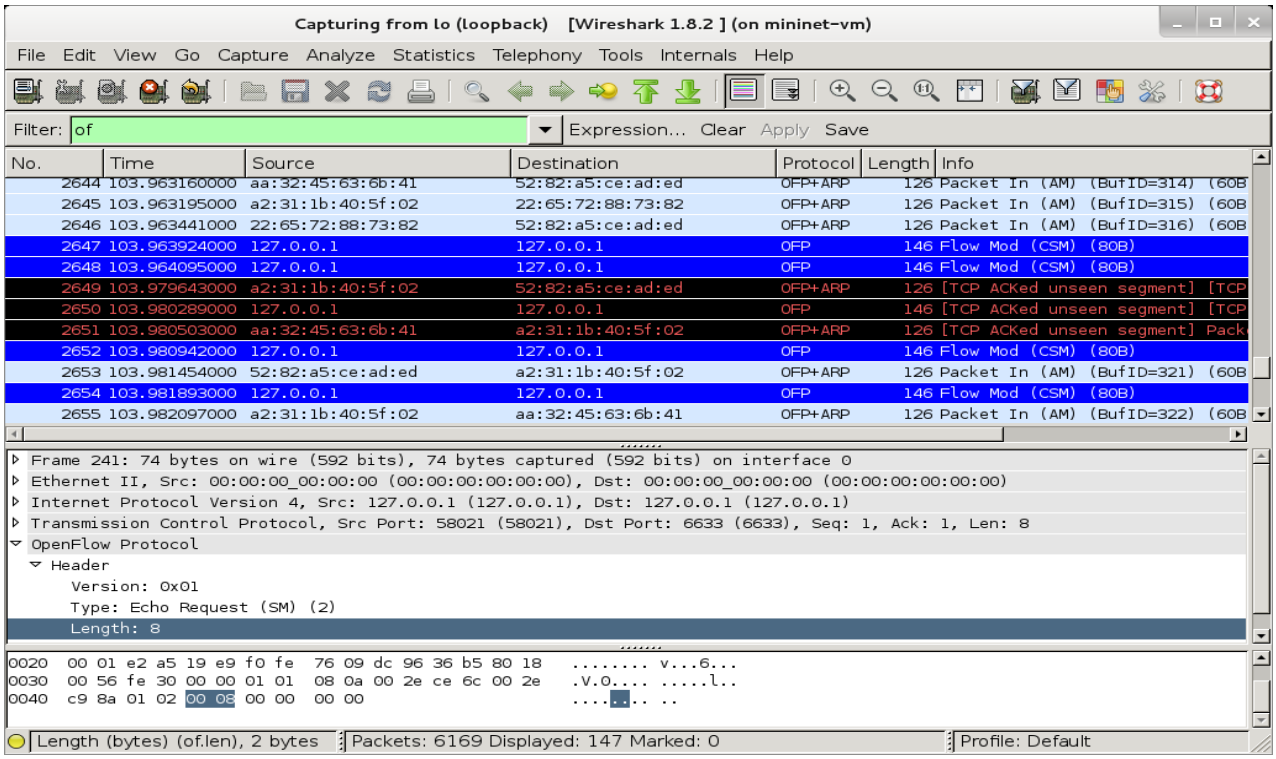

Figure 5-7: Monitoring Network using Wireshark

CHAPTER 6

Conclusion and Future Work

"Some people suggest that SDN and Virtualization is same. This is wrong as SDN is merely a method to achieve network virtualization and both of them cannot be taken to be equal. SDN has variety of other features like analysing, monitoring and increasing security"

Software Defined Networking has lots of advantages as it enables the user to control the network which in turn enables us to optimize the speed, performance and security of the Network. There are lots of aspects to be looked upon before SDN is adopted for universal interoperability. This is because software implementation doesn't guarantee results in hardware. But there are no doubts that SDN is evolving faster every day and it's not long before we see network vendors accepting it universally.

As in this thesis, a method was developed to address the latest security concerns of most network operators i.e. ARP Spoofing. Similarly, as we can monitor the network traffic in wireshark, we can use it to optimize the path of data flow and similarly there are various other applications of SDN.

But major limitation of the emulator used here i.e. Mininet is that it can emulate networks using slower links and it is not efficient for high speed links, it is because packets are forwarded by a collection of software switches that share CPU and memory resources and usually have lower performance than dedicated switching hardware. As a result the whole idea has till now only been limited to research work and there still loopholes that have to be covered during this phase.

6.1 SDN Myths

- Some people are of the opinion that purpose of SDN is only to reduce costs. This is not the case as SDN will enable users to automate the network as well as optimize by controlling the flow of packets.

- Some have opinion that SDN will compromise security. This a common fear of all IT professionals as new technology like that of SDN can scare people but once SDN is enabled with proper security protocols, it can increase the overall security of the network.
- Some think that SDN will hamper the hardware market. But in reality we are merely controlling the hardware devices with a software and not "replacing" it. Hence this is a wrong notion.

- Some people suggest that SDN and Virtualization is same. This is wrong as SDN is merely a method to achieve network virtualization and both of them cannot be taken to be equal. SDN has variety of other features like analysing, monitoring and increasing security.
- People think that SDN is going to take over the market very quickly. This is totally false as vendors have been apprehensive about implementing SDN due to a variety of reasons and hence we cannot see SDN capturing the market unless all these issues are addressed.

Scope for Future Work

Since the concept of SDN and Network virtualization is new and evolving quickly, there are lots of scope to work further in this field. For a beginner, an immediate area can be optimizing the speed of the network using SDN controller. This can be done by programming the pox controller in the mininet VM as was done for ARP prevention.

Another area that can be worked upon is the Network Virtualization part, with a good knowledge of DBMS the concept of FlowN can be implemented which will solve the major hurdles for obtaining interoperability between the different network infrastructures.

Another aspect that can be looked upon is debugging of the Software defined Network. As it is a software after all and all software have bugs which is the major reason why SDN has not been accepted universally, so debugging will help in building trust and making the concept of SDN more robust.

References

[1] "Shaheed Bhagat Singh State Technical Campus," [Online]. Available: http://sbsstc.ac.in/icccs2014/Papers/ProceedingsICCCS2014.pdf. [Accessed January 2015].

[2] "TREMA," 24 September 2012. [Online]. Available: http://www.trema.info/2012/09/repeater-hub/.

[3] V. R. a. S. Nandi, "Detecting ARP Spoofing: An Active Technique," *LNCS Publication ,* 2012.

[4] "OpenFlow," [Online]. Available: http://www.openflow.org/wk/index.php/OpenFlow_Tutorial. [Accessed 2015].

[5] H. K. a. N. Feamster, "Improving network management using SDN," *IEEE Communications Magazine,* pp. 114-119, 2013.

[6] K. Bakshi, "Considerations for Software Defined Networking(SDN):Approaches and Use Cases," *IEEE Aerospace Conference,* 2013.

[7] Open Networking Foundation(ONF) , "Software-Defined Networking: The New Norm for Networks," *White Paper,* 2012.

[8] I. S. a. T. T. Leadership, *Software Defined Networking: A new paradigm for virtual, dynamic, flexible networking,* October, 2012.

[9] H. K. a. N. Feamster, "Improving network management using SDN," *IEEE Communications Magazine,* pp. 114-119, February 2013.

[10] E. K. J. R. Dmitry Drutskoy, "Scalable Network Virtualization in Software-Defined Networks," *IEEE Communications Magazine,* February 2013.

[11] "http://bigswitch.com," [Online]. Available: http://bigswitch.com/sites/default/files/sdn_resources/onf-whitepaper.pdf?. [Accessed September 2014].

[12] D. Drutskoy, 2012. [Online]. Available: https://www.google.co.in/search?q=flowN+and+flowVisor.&biw=1366&bih=623&source=lnms&sa=X&ei=L7NPVeSeIomMuATmqYCADw&ved=0CAYQ_AUoAA&dpr=1#.